powershell scripts

- New Page
- New Page
- New Page

New Page

```
<#
Praktijkleren Examenopdracht – PowerShell
Onderwerp: Kentekencheck automatiseren
Auteur: <jouw naam>
Datum: <datum>


Dit script automatiseert het opzoeken van voertuiggegevens
via RDW Open Data.
#>


function Vraag-Kenteken {
    <#
    Vraagt maximaal 3 keer om een geldig kenteken.
    Geeft het kenteken of $null terug.
    #>
    $pogingen = 0  # variabele


    while ($pogingen -lt 3) {  # lus met limiet
        $kenteken = Read-Host "Voer een kenteken in (6 tekens)"
        $kenteken = $kenteken.ToUpper().Replace("-", "").Trim()


        if ($kenteken.Length -eq 6) {  # beslissing
            return $kenteken
        }
        else {
            $pogingen++
            Write-Host "Ongeldig kenteken. Poging $pogingen van 3." -ForegroundColor Red
        }
    }


    Write-Host "Te veel foutieve pogingen. Programma stopt." -ForegroundColor Yellow
    return $null
}
```

```
function Haal-VoertuiggegevensOp {
    param (
        [string]$Kenteken
    )

    $url = "https://opendata.rdw.nl/resource/m9d7-ebf2.json?kenteken=$Kenteken"

    try {
        return Invoke-RestMethod -Uri $url -Method Get
    }
    catch {
        return $null
    }
}

function Toon-Gegevens {
    param (
        $Data
    )

    if (-not $Data) {  # beslissing
        Write-Host "Geen voertuiggegevens gevonden." -ForegroundColor Yellow
        return
    }

    $voertuig = $Data[0]
    Write-Host ""
    Write-Host "Voertuiggegevens:"
    Write-Host "Kenteken:" $voertuig.kenteken
    Write-Host "Merk:" $voertuig.merk
    Write-Host "Type:" $voertuig.handelsbenaming
    Write-Host "Voertuigsoort:" $voertuig.voertuigsoort
    Write-Host "Eerste toelating:" $voertuig.datum_eerste_toelating
}

# Hoofdprogramma
$doorgaan = $true  # variabele

while ($doorgaan) {  # lus
    $kenteken = Vraag-Kenteken
```

```
    if ($null -eq $kenteken) {
        break
    }


    $gegevens = Haal-VoertuiggegevensOp -Kenteken $kenteken
    Toon-Gegevens -Data $gegevens


    $keuze = Read-Host "`nNog een kenteken controleren? (j/n)"
    if ($keuze.ToLower() -ne "j") {  # beslissing
        $doorgaan = $false
    }
}


Write-Host "Programma afgesloten."
```

New Page

```
<#
Praktijkleren Examenopdracht – PowerShell
Onderwerp: Subnetcalculator
Auteur: <jouw naam>
Datum: <datum>


Dit script berekent netwerkgegevens op basis van
een IP-adres en subnetmasker (CIDR).
#>


function Vraag-IPAdres {
    $pogingen = 0  # variabele


    while ($pogingen -lt 3) {  # lus met limiet
        $ip = Read-Host "Voer een IP-adres in (bijv. 192.168.1.10)"
        $delen = $ip.Split(".")


        if ($delen.Count -eq 4 -and ($delen | ForEach-Object { $_ -match '^\d+$' -and [int]$_ -le 255
})) {
            return $ip
        }
        else {
            $pogingen++
            Write-Host "Ongeldig IP-adres. Poging $pogingen van 3." -ForegroundColor Red
        }
    }


    Write-Host "Te veel foutieve pogingen." -ForegroundColor Yellow
    return $null
}


function Vraag-Subnetmasker {
    $cidr = Read-Host "Voer subnetmasker in (CIDR, bijv. 24)"
```

```
    if ($cidr -match '^\d+$' -and [int]$cidr -ge 0 -and [int]$cidr -le 32) {
        return [int]$cidr
    }
    else {
        Write-Host "Ongeldig subnetmasker." -ForegroundColor Red
        return $null
    }
}


function Bereken-Hosts {
    param (
        [int]$CIDR
    )


    $hostBits = 32 - $CIDR
    $hosts = [math]::Pow(2, $hostBits) - 2
    return [int]$hosts
}


# Hoofdprogramma
$ip = Vraag-IPAdres
if ($null -eq $ip) { return }


$cidr = Vraag-Subnetmasker
if ($null -eq $cidr) { return }


$aantalHosts = Bereken-Hosts -CIDR $cidr


Write-Host "`nResultaat:"
Write-Host "IP-adres:" $ip
Write-Host "Subnetmasker: /$cidr"
Write-Host "Aantal mogelijke hosts:" $aantalHosts
```

New Page

```
<#
Praktijkleren Examenopdracht – PowerShell
Onderwerp: Logbestand analyseren
Auteur: <jouw naam>
Datum: <datum>


Dit script analyseert een logbestand en telt
het aantal foutmeldingen.
#>


function Lees-Logbestand {
    param (
        [string]$Bestandsnaam
    )


    if (Test-Path $Bestandsnaam) {
        return Get-Content $Bestandsnaam
    }
    else {
        Write-Host "Bestand niet gevonden." -ForegroundColor Red
        return $null
    }
}


function Tel-Fouten {
    param (
        [string[]]$Regels
    )


    $foutTeller = 0  # variabele


    foreach ($regel in $Regels) {  # lus
        if ($regel.Contains("ERROR")) {  # beslissing
            $foutTeller++
```

```
      }
   }


   return $foutTeller
}


# Hoofdprogramma
$bestandsnaam = "logbestand.txt"  # variabele


$regels = Lees-Logbestand -Bestandsnaam $bestandsnaam
if ($null -eq $regels) { return }


$fouten = Tel-Fouten -Regels $regels


Write-Host "`nAnalyse resultaat:"
Write-Host "Aantal regels in bestand:" $regels.Count
Write-Host "Aantal foutmeldingen:" $fouten
```